Automated Testing System for Android Applications based on Dynamic Taint Propagation

Rakesh M
M.Tech 2nd semester
Department of CSE
City engineering college

Mukesh Kamath
Associate professor
Department of CSE
City Engineering College

Abstract: **As the increasing downloads of applications via Android Platform, more and more malicious codes were injected in those applications. And some problems are caused by that malicious code such as economic loss and privacy issues. Android has the highest market share of smartphone operating system, the security of Android platform is extremely important. Therefore, the security testing and evaluation of applications is imperative. Dynamic taint propagation is the most common method to do the test, but there are two problems: a) If the custom ROM runs in the smartphone, the running speed of ROM will be limited to the smartphone's battery life and computing power. b) If the program was running in emulator in PC, the efficiency will be very poor because of the manual operation for the triggering action during the running time. The paper presents an automated testing method which was accomplished in emulator. In addition, the system will record the tree structure of Activity and control distribution of each Activity. The test results showed that the system can trigger all the controls and compared with manual test, this method was proven to be more effective and completely.**

Keywords: Android Platform, applications, dynamic taint propagation, automated testing, emulator.

1 INTRODUCTION

According to Jump tap, Android occupied 58.8% of the mobile market in 2012 and the share was increasing [1].Corresponds with the growth of Android's market share, the number of applications for android had an explosive growth. Google announced that the download of its electronic products has exceeded 25 billion and the applications have exceeded 675 thousand [2]. What came along with the increasing were lots of problems. Lots of malicious codes were injected to the hot applications to execute malicious behaviors and the behaviors led to great security threats. Common results of malicious behaviors are divided into two kinds: economic loss and privacy leakage. The first case is economic loss. In this case, the application that was injected by malicious code will order fee-based service via message, or make calls, or surf the internet in a covert way and lead to economic loss. The second case is privacy leakage. In this case, the application that was injected by malicious code will read the privacy Information such as contact information, location, device number, messages and schedule without telling the user, then the information will be delivering to the attacker via the internet or message. Worse, the malicious code can control the smartphone to monitor the user. According to "the report of the security situation of Chinese mobile phone in 2012" from 360 security center [3], there were 174977 new malware based on mobile in 2012 with a growth of 1907 percent year-on-year. Among the malware, there were 123681 new samples that based on Android and they occupied 71% of all the new samples. During the results of those malware, economic loss takes the top spot with 52 percent and the next is privacy leakage with 28 percent. Therefore, it is very important to test the security of applications based on Android.

2 CURRENT STUDY AND INSUFFICIENCY

According to the study [4-9] of malicious code in the application, the paper gives a classification; it is showed inFigure 1.

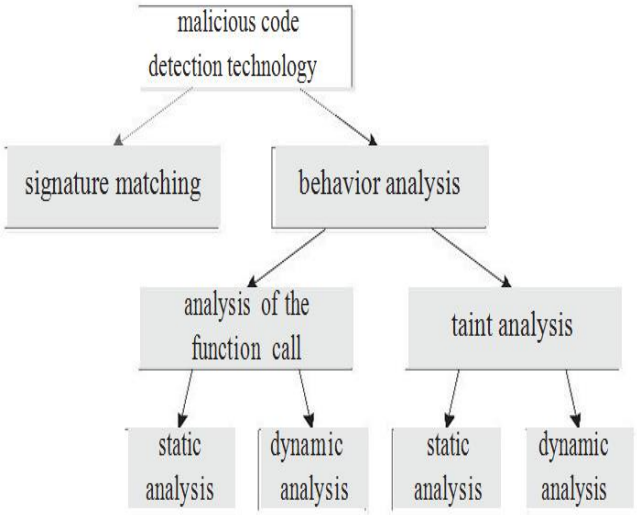

Figure 1. Classification of Malicious Code Detection Technology.

Enck [10] put forward a research method named dynamic taint analysis and had accomplished the design and implementation of Taint Droid. TaintDroid is a real-time detection system based on the analysis of dynamic information flow. The system modifies Android's application framework and Dalvik virtual machine, and it supports four levels' data tracking: Variable level, method level, message level and file level. In the system installed TaintDroid, tags will be attached to the data that came from the privacy data sources and they will spread with the data in the system. When the privacy data with those tags are sent through messages or internet, a warning will be triggered and this event will be recorded in logs Because of the good performance that Taint Droid demonstrated in the aspect of detecting malicious behaviors(especially in the aspect of privacy leakage), more and more projects are based on Taint Droid. But there is a problem: the original Android system was modified when Taint Droid was installed, this will cause problems in the aspect of stability and efficiency, so applying Taint Droid in mobile devices is not appropriate.

In addition, TaintDroid also can be applied in emulator, but the application needs some more manual operations and the efficiency is low, and there will be some omissions. Mr.Hu [11] proposed a method that send random data to the application to trigger the application, the problem is that there will be no relationships between the triggered events. Aiming at the above problems, the paper put forward an automated testing scheme. In this scheme, applications will be tested in Android emulator and controls of all Activity in the applications will be triggered automatically. Compared with manual testing, this scheme is more efficient and the coverage is higher.

3 SOME AUTOMATED TESTING TOOLS

The Android SDK provides some tools to aid developers in automated testing work; they are Monkey, MonkeyRunner and HierarcyViewer.

Monkey, it is a command line tool in PC. It can be used in the Android emulator or a real device, generating pseudo random event stream and simulating a click, touch, gesture, and system events. Its format is as follows:

adb shell monkey [options] The disadvantage of Monkey is that its test operation is random and cannot be able to customize, so that it has some limitations. MonkeyRunner, it provides a set of APIs to control the emulator or device. Developers can write Python script to call the APIs for installing and running Android applications, and pressing key or entering text in applications. But for different applications, MonkeyRunner needs to write different Python scripts for testing, workload is too big. HierarchyViewer, it is a tool designed to help developers debug and optimize UI which is provided by the Android. Through this tool, developers can get the window list of application which is running on the device, and the control hierarchy information of interface. But HierarchyViewer can only view the control structure of current Activity, when the Activity changes, it cannot be updated in real time. In conclusion, the three tools provided by the Android SDK are not effective for automated testing.

4 IMPLEMENTATION OF AUTOMATED TESTING SYSTEM

4.1 The Framework of Automated Testing System

Automated testing system, running in the host interacts with the applications which run in the emulator by sockets. The overall frame structure is shown in Figure 2.In the figure, ViewServer, provided by Android, is a service program for the convenience of developers to debug interface. By using ViewServer, the developers will be able to get the window list of the application which is running in the emulator, and the interface information of each window.The Automated test system mainly includes two modules: interface analysis and interface interaction. The main function of interface analysis module is to establish a connection with ViewServer and get the window list and control information of the interface through the command interface provided by ViewServer. Then they can get the existing triggers on the interface after further analysis. The interface interaction module bases on control information of interface analysis module to generate the control triggering events according to certain order and then send interactive commands to the application.

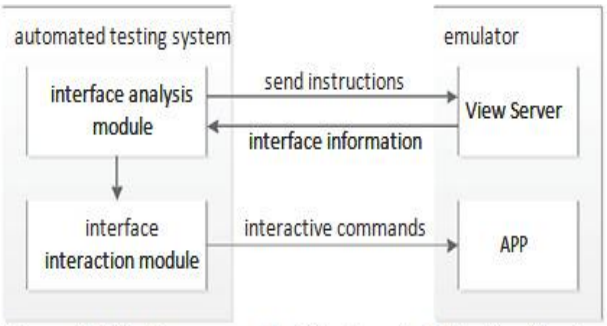

Figure 2. The Framework of Automated Testing System.

4.2 Interface Analysis

Application usually contains multiple interfaces, and as they progress through the interaction, the program will jump between each interface. In order to analysis to all of the interfaces, interface analysis module records the hierarchy of the interface, and uses recursion method to analyze each interface and their interaction. Activity is the basic interface of Android system. Its call hierarchy information can use the Activity call graph representation, as shown in Figure 3. Generally every Android application has a main interface as the entrance to the program, for example Main Activity which is shown in the figure. The program can jump to other interfaces by triggering Controls

on the main interface. In the figure, the Activity such as Activity1, Activity2 and Activity3, Constitutes the second layer of Activity call graph, under which there may be another layer of activity, and thus these Activities form a tree structure.The call graph of Activity is gradually established in the process of the whole work in automated test system. After each complete control trigger, Interface analysis module can obtain a list of Activity information in the emulator from ViewServer through the command. If a new Activity is found, interface analysis module will add it to the Activity call graph, as a child interface of the previous Activity.

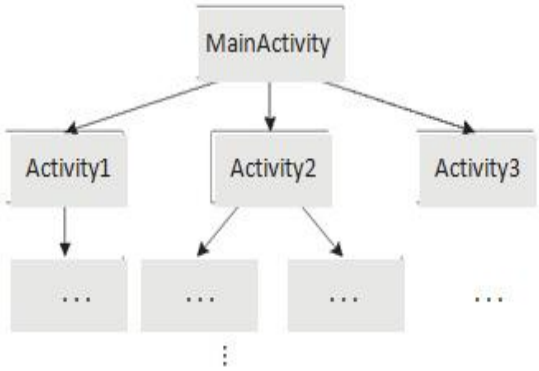

Figure 3. The Activity Call Graph.

When entering a new Activity, Interface analysis module will obtain the layout of control of the current interface and Carry on the analysis. The returning information of ViewServer contains the type, coordinates, width and margin attribute information of each control.Interface analysis module parses all information of the control and classifies in accordance with the control types. One kind is editable text control and another kind is clickable control. In Android system, the editable text control contains EditText and AutoCompleteTextView and the clickable control contains Button, ImageButton, ToggleButton, RadioButtonCheckBox, and so on. Interface analysis module can carry out the analysis of the control and compute the control center coordinates, according to the margin and width information of the control, so that the Interface interaction module works when the control triggers.

The basic attributes of each control include the width, height, left margin top margin, respectively is layout_width, layout_height, mLeft and mTop. According to these properties, the center coordinate of the control can be calculated. The center coordinates (x_center, y_center) is shown below.

x_center = mLeft + layout_width/2 y_center = mTop + layout_height/2

But mLeft and mTop, respectively is the margin of the current conrtol relative to its parent control. So x_center and y_center need to add the relative coordinates of the left and top margins of its parent control, until to the root node. The pseudo code is shown as follows.

```
while (! rootNode)

{

x_center += mLeft; y_center += mTop;

}
```

After getting the center coordinate, the system uses adb shell to simulate click event. A Button, for example, if its center coordinates is (40, 200), the click code is shown as follows

Adb shell sendevent /dev/input/event0 3 0 40 adb shell sendevent /dev/input/event0 3 1 200

Adb shell sendevent /dev/input/event0 1 330 1 adb shell sendevent /dev/input/event0 0 0 0

Adb shell sendevent /dev/input/event0 1 330 0 adb shell sendevent /dev/input/event0 0 0 0

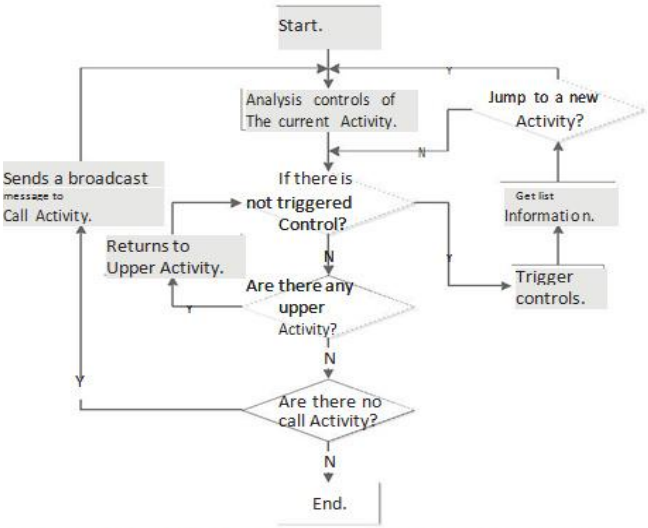

Figure 4. The Workflow of Automated Testing System.

4.3 Interface Interaction

In order to let the malicious act hidden in application enforce as much as possible, automated interactive engine requires all the Activity invoked, and triggers all the controls on the interface as comprehensive as possible. There are two main types of call way for Activity in Android system. One calls through triggering controls on the interface, and another calls through radio message of the Android system. Considering the two cases, the interface interaction module first calls the Activity which can be invoked, by triggering interface controls. Then after all the invoked controls are triggered, the interface interaction module will query whether all the Activity

stated in the AndroidManifest.xml file are invoked. If there is any, the interface interaction module will generate radio news to call, according to the trigger condition stated by Activity, and then it will analysis and trigger all the controls of the Activity. As to control trigger order, the general design of the program will allow users to input text messages first and then click on the button. So, after getting the control information of interface analysis module, the interface interaction module of automated interactive engine will first the trigger the click and input events for text control, and then trigger the click event for the button. The workflow of automated testing system is shown in Figure 4.

5 TESTING

5.1 Testing Environment

The testing environment is shown in Table 1.

5.2 Testing Method

Testing purpose is to prove that automated testing system can complete the operations of all the controls in the applicationthe most intuitionistic testing method is observing the application interfaces. But it's subjective and cannot quantitative, so in this paper, we put forward a kind of objective testing method. We chose 10 popular applications and 10 new applications from an app store. Then respectively use human interaction and automated testing system for application testing. Based on a platform named Droid Box, and the testing results is shown in Table 2.

Name	Parameter
CPU	Intel® Core™ i7-2600K 3.40GHz
RAM	16.00 GB
Virtual machine	VMware® Workstation 9.0.1
OS	Ubuntu 12.04 LTS 32bit
Android version	Android 2.3.3

Table 1. The testing environment.

Privacy type	Human interaction	Automated testing	Scale
IMSI	6	6	15%
IMEI	18	18	45%
ICCID	4	4	10%
Phone number	2	2	5%
Location	4	4	10%
Contacts	0	1	2.5%
SMS	1	1	2.5%

Table 2. The testing results of applications.

5.3 Testing Results and Analysis

From the table, we can see that compared to human interaction and automated testing system, the results are basically identical. The leak of contacts in an application has not been detected in human interaction, but in the automated testing system it has been found. The cause of this situation is that a button has not been triggered in human interaction.As can be seen from the testing results, compared to human interaction, the test completeness of the automated testing system has been well-documented. When the number of applications rises sharply, the advantages of the automated testing system will be better manifested. That is, when testing the same amount of applications, time is rarely used by the automated testing system needs, and efficiency is improved greatly. More than that, it can also avoid the undetected controls or Activities in human interaction.

6 CONCLUSION

This paper puts forward and implements the automated testing system, which is applied in the Android emulator and combined with dynamic taint propagation. It greatly improves the testing efficiency and coverage compared to previous way of human interaction. Not only to the current Android security research it provides the reference, but also can be reference by other mobile intelligent terminal OS.

REFERENCES

[1]Jumptap,"Android and iPhone Now Hog 91% of Mobile OS Market Share", 2013, available online from: http://www.jumptap.com/blog/android-and-iphone-now-hog-91-of-mobile-os-market-share

[2]91 Wireless,"The report of mobile application development trend in Q3 of 2012", 2012, available online from: http://www.eguan.cn/download/zt.php?tid=1167

[3]360 Security Center,"The report of the security situation of Chinese mobile phone in 2012", 2012, available online from: http://shouji.360.cn/securityReportlist/securityReport_9 5.html

[4]N. Idika, and A. P. Mathur,"A survey of malware detection techniques," Department of Computer Science, Purdue University, Tech. Rep., 2007. [5]Vinod P., and V. Laxmi, M. S. Gaur,"Survey on malware detection methods," 3rd Hackers' Workshop on Computer and Internet Security, March 17-19, 2009, pp.74-79.

[6]G. Hu, and D. Venugopal,"A malware signature extraction and detection method applied to mobile networks," Proceedings of the 26th IEEE International Performance Computing and Communications Conference, 2007, pp.19-26. [7]D. Venugopal, and G. Hu,"Efficient signature based malware detection on mobile devices," Mobile Information Systems, 2008, 4(1), pp.33-49.

[8]A.-D. Schmidt, F. Peters, F. Lamour, and S. Albayrak,"Monitoring Smartphones for Anomaly Detection," in MOBILWARE 2008, International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, Innsbruck, Austria, 2008, pp.92-106.

[9]M. Chandramohan, and H. Tan,"Detection of Mobile Malware in the Wild," Volume:PP, Issue:99, IEEE Early Access, 2012, 45(9), pp.65-71.

[10]W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth,"TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," Proceedings of the 9th USENIX conference on Operating systems design and implementation, Berkeley, CA, USA: USENIX Association, 2010, pp.1-6.

[11] C. Hu, I. Neamtiu,"Automating GUI testing for Android applications," Proceedings of the 6th International Workshop on Automation of Software Test, New York, NY, USA: ACM, 2011, pp.77-83.